Introduction

Given a folder containing a sequence of images, the problem is to compress all these images into a single file. Then, the program extracts and shows the compressed images from the file. The method for the program is an altered version of a Set Redundancy Compression method. The program checks through all each pixel of all images in the folder and takes note of each pixels highest and lowest value. With the minimum and maximum values in hand, the program proceeds to compress through each images.

Rationale

Why this method?

Initial idea for this problem is to merely use RLE method to compress each images in the folder. However, Set Redundancy Compression method’s advantages is that it can exploit repeating pixel values throughout the images. There are 3 known methods for Set Redundancy Compression and that is using the differential, the predictive and the centroid method. The second idea is to compress the image via predictive method. With predictive method, the program needs to calculate the levels of the image. The program manually calculated the levels for the first row and column of the pixel and proceeded to use MMP3 method of prediction with the rest. Then, the program uses this levels to create the predicted pixel. With the predicted pixel value created, the program only needs to take the difference of the subtracted and of the original to create the compressed version of the image. The problem of the second idea is the problem of decompressing the image when the only value available is the difference between the original and predicted value. The third idea then is to alter the MMP method and instead of calculating the levels of the pixel, the program proceeds to only change the range of the maximum and minimum values. The plan now is to reduce the variety of colors used by the image, culling values higher than the altered max and min.

Background

Why this method was created? Who created it? What is the pros and cons of using it

Set redundancy is a concept established by Karadimitrou in 1996. The compression algorithms existing at the time only recognizes the redundancy in the coding, spatial, and psychovisual. Coding redundancy simply refers to the repeated information in the coding of an image, an example of this is reducing an image to the least amount of coding to represent all values necessary. Spatial or the interpixel redundancy refers to the relationship between the current pixel and its neighbors, predicting the value of the pixel given its neighbor is a method of image compression. Psychovisual redundancy refers to the minute details of the image not necessary for a person to perceive the entire image. As human vision cannot be as accurate as the computer’s quantitative analysis of an image, simply reducing details can be a form of image compression.

Karadimitrou offers a 4th type of redundancy, the Set Redundancy. With this, they acknowledge the presence of inter-image redundancy, defining the similar images as those that have (a) similar pixel intensities in similar areas, (b) similar histograms, (c) similar edge distributions, and (d) similar distributions of features. They then proceed to list the different methods of Set Redundancy Compression like the Min-Max Differential, the Min-Max Predictive, The Centroid and other methods.

The advantages of the Set Redundancy Compression is its ability to take advantage of similar images whereas previous methods compresses individual images. This method is best applied to sequences of images or in Medical imaging where large amounts of relatively similar images are produced every day, thus there is a need for a compressed image. Like other compression methods, Karadimitrou’s concept only resolves the set redundancy inherent in similar images thus it requires and is mostly usable when applied to such images.

Methodology

The program initially requires the folder where images will be compressed. The program then prepares two arrays that holds the maximum and minimum values each pixels has managed to reach throughout all images. With the maximum and minimum values noted, the program now has a range of values a pixel can possibly be. The program, then proceeds an altered version of the Set Redundancy Compression Method where it will shift the color of the pixel down a certain value. This is needed to prevent compressed image from reaching its higher values and becoming heavier in the process. An image with dimmer and darker values turns out to be lighter than an image with brighter values.

When a compressed version of an image is ready, the program proceeds to append it to an array that collects all compressed images. Once the images in the folder are exhausted, the program proceeds to create an .cmp file of the variable via pickle.dump method. This .cmp file now contains all the compressed image from the folder. The next step is for the program to read the .cmp file, convert each of its array of compressed images into actual readable version and to then present them to the user. The program also showcases the original and the compressed file sizes and its compression ratio as well as the length of time it took for the program to compress the images.

Results and Discussion

What is the output? How fast does it perform? Is this the result you expected?

There are two types of outputs created in the process of compression and extraction of an image. First is the compiled file containing all images, this .cmp file is much heavier than its original images. In a sample run, 20 800x450 pixel image with a total file size of 1.65 mb can create a .cmp file reaching 150 mb. This .cmp file will still undergo extraction where the actual compressed images are lighter than the original.

Interestingly, since the program intends to cull pixels with values higher than its altered max value, images should have dimmer and darker images. However, if a sequence of image contains pure consistent white pixel all throughout, its compressed version turns out to be heavier. This only occurs to simple images as compression are successful when applied to more complex samples. The hypothesized explanation regarding this situation is that there is not much change between images thus when run through the program’s altered set redundancy compression algorithm, there is not much to compress, thus the image only became a heavier version than the expected result.

Conclusion

How was the experience? Is this a recommended technique? Things that can be improved?

Successfully compiling several images in 1 go can be a very difficult challenge without the set redundancy method. While the program did not follow any of the established methods, it still managed to compress several sequences of pictures successfully. Merely deducting the pixel values of the image to create a compressed version of it is a basic way of compression, it would be better if the actual methods of compression, that is the MMP, MMD and the centroid methods are emulated. However, because there is lack of relevant resource in decompressing this data, any hopes of compressing these images are difficult. Using the algorithm employed in this program is not recommended so instead it would be best to follow the more established methods of Set redundancy compression. Furthermore, the current algorithm does not take full advantage of its maximum and minimum array of values.

References

**https://www.ques10.com/p/7293/what-are-different-types-of-redundancies-in-digi-1/#:~:text=Interpixel%20redundancy%20is%20due%20to,is%20they%20are%20highly%20correlated.**

Ait-Aoudia, S., & Gabis, A. (2006). A comparison of set redundancy compression techniques. *EURASIP Journal on Advances in Signal Processing*, *2006*, 1-13.

Karadimitriou, K. (1996). *Set redundancy, the enhanced compression model, and methods for compressing sets of similar images*. Louisiana State University and Agricultural & Mechanical College.